# APPENDIX B

# FOR

# UNITED STATES LETTERS PATENT

| | |
|---|---|
| TITLE: | LINKS FOR CONFIGURING A VIRTUAL PRIVATE NETWORK |
| APPLICANT: | MATTHEW W. POISSON<br>MELISSA L. DESROCHES<br>JAMES M. MILILLO |

16 PAGES

"EXPRESS MAIL" Mailing Label Number EL182578382

Date of Deposit April 2, 1999

I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as **"Express Mail Post Office To Addressee"** with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

[illegible]

Alison Brazil

Appendix 13

```
// Obtain all information from the Extranet Switch
// The result will then be "imported" into the OCM
// product.
// The "back" statements are used to symbolize the
// end of a section, it also makes it easier to import
Error=Error

// Obtain info ExtranetSwitch Basic Tab
"ExtranetDevice.IP_ADDR.IP_ADDR "
call omget using {"dns.systemipaddress"}
"\nExtranetDevice.HOSTNAME.HOSTNAME "
call omget using {"dns.systemname"}
"\nExtranetDevice.SWITCH_TYPE.SWITCH_TYPE "
call omget using {"flash.ModelNumber"}
"\nExtranetDevice.CUR_VERSION.CUR_VERSION "
call omget using {"DirRestore.CurVersion"}
"\nExtranetDevice.DOMAIN_NAME.DOMAIN_NAME "
call omget using {"dns.domainname"}
"\nExtranetSwitch.PRIMARY_SERVER.PRIMARY_SERVER "
call omget using {"dns.primarydnsserver"}
"\nExtranetSwitch.SECONDARY_SERVER.SECONDARY_SERVER "
call omget using {"dns.secondarydnsserver"}
"\nExtranetSwitch.TERTIARY_SERVER.TERTIARY_SERVER "
call omget using {"dns.tertiarydnsserver"}

// Obtain info for Shutdown Tab
"\nExtranetSwitch.DISABLE_NEW_LOGINS.DISABLE_NEW_LOGINS "
call omget using {"Security.NewLoginsEnabled"}
"\nExtranetSwitch.DISABLE_RESTART_LOGINS.DISABLE_RESTART_LOGINS "
call omget using {"Shutdown.DisableLoginsOnRestart"}
"\nExtranetSwitch.SYSTEM_SHUTDOWN.CHOICEBOX "
call omget using {"Shutdown.Mode"}
"\nExtranetSwitch.SYSTEM_SHUTDOWN.TEXTBOX "
call omget using {"Shutdown.EventTimeDelay"}
"\nExtranetSwitch.POST_SHUTDOWN.POST_SHUTDOWN "
call omget using {"Shutdown.EventAction"}
"\nExtranetSwitch.REBOOT_DRIVE.REBOOT_DRIVE "
call omget using {"DiskRdn.BootDevice"}

// lets get some capacity stuff
"\nExtranetSwitch.TUN_USERS.NUM_USERS "
call omget using {"dbgroups.group[ROOT::SUBTREE].persons.numentries"}
"\nExtranetSwitch.TUN_USERS.MAX_TUNNELS "
call omget using {"Flash.maximumusers"}

// Obtain info for ExtranetSwitch  Admin tab
"\nExtranetSwitch.USER_ID.USER_ID "
call omget using {"flash.adminuid"}
"\nExtranetSwitch.PASSWORD.PASSWORD "
call omget using {"flash.adminpassword"}
"\nExtranetSwitch.IDLE_TIMEOUT.IDLE_TIMEOUT "
call omget using {"DbGroups.Group[ROOT].Accounts.Account[GENERAL,-
].AdminIdleTimeout"}

// Obtain info for ExtranetSwitch Service Tab
"\nExtranetSwitch.IPSEC.PUBLIC "
call omget using{"security.untrustedipsecenabled"}
```

```
"\nExtranetSwitch.IPSEC.PRIVATE "
call omget using{"security.trustedipsecenabled"}
"\nExtranetSwitch.PPTP.PUBLIC "
call omget using{"security.untrustedpptpenabled"}
"\nExtranetSwitch.PPTP.PRIVATE "
call omget using{"security.trustedpptpenabled"}
"\nExtranetSwitch.L2TP_L2F.PUBLIC "
call omget using{"security.untrustedl2fenabled"}
"\nExtranetSwitch.L2TP_L2F.PRIVATE "
call omget using{"security.trustedl2fenabled"}
"\nExtranetSwitch.L2TP_L2F.PUBLIC "
call omget using{"security.untrustedl2tpenabled"}
"\nExtranetSwitch.L2TP_L2F.PRIVATE "
call omget using{"security.trustedl2tpenabled"}
"\nExtranetSwitch.HTTP_PRIVATE.HTTP_PRIVATE "
call omget using{"security.trustedhttpenabled"}
"\nExtranetSwitch.SNMP_PRIVATE.SNMP_PRIVATE "
call omget using{"security.trustedsnmpenabled"}
"\nExtranetSwitch.FTP_PRIVATE.FTP_PRIVATE "
call omget using{"security.trustedftpenabled"}
"\nExtranetSwitch.TELNET_PRIVATE.TELNET_PRIVATE "
call omget using{"security.trustedtelnetenabled"}
"\nExtranetSwitch.ALLOW_T2T.ALLOW_T2T "
call omget using{"security.allowtunneltotunnel"}
"\nExtranetSwitch.ALLOW_EUTBO.ALLOW_EUTBO "
call OmGet using {"Security.AllowClientToBranch"}
"\nExtranetSwitch.ALLOW_BOTBO.ALLOW_BOTBO "
call OmGet using {"Security.AllowBranchToBranch"}


// Obtain info ExtranetSwitch AutoBackup Tab
"\nExtranetSwitch.ABUG_ROW1.ABUG_ENABLED "
call omget using {"dirbackup.primaryzenabled"}
"\nExtranetSwitch.ABUG_ROW1.ABUG_HOST "
call omget using {"dirbackup.primaryhost"}
"\nExtranetSwitch.ABUG_ROW1.ABUG_PATH "
call omget using {"dirbackup.primarypath"}
"\nExtranetSwitch.ABUG_ROW1.ABUG_INTERVAL "
pint=call omgetnum using {"dirbackup.primaryinterval"}
pint=pint/60
pint
"\nExtranetSwitch.ABUG_ROW1.ABUG_USERID "
call omget using {"dirbackup.primaryusername"}
"\nExtranetSwitch.ABUG_ROW1.ABUG_PASSWORD "
call omget using {"dirbackup.primarypassword"}
"\nExtranetSwitch.ABUG_ROW2.ABUG_ENABLED "
call omget using {"dirbackup.secondaryzenabled"}
"\nExtranetSwitch.ABUG_ROW2.ABUG_HOST "
call omget using {"dirbackup.secondaryhost"}
"\nExtranetSwitch.ABUG_ROW2.ABUG_PATH "
call omget using {"dirbackup.secondarypath"}
"\nExtranetSwitch.ABUG_ROW2.ABUG_INTERVAL "
sint=call omgetnum using {"dirbackup.secondaryinterval"}
sint=sint/60
sint
"\nExtranetSwitch.ABUG_ROW2.ABUG_USERID "
call omget using {"dirbackup.secondaryusername"}
```

```
"\nExtranetSwitch.ABUG_ROW2.ABUG_PASSWORD "
call omget using {"dirbackup.secondarypassword"}
"\nExtranetSwitch.ABUG_ROW3.ABUG_ENABLED "
call omget using {"dirbackup.tertiaryzenabled"}
"\nExtranetSwitch.ABUG_ROW3.ABUG_HOST "
call omget using {"dirbackup.tertiaryhost"}
"\nExtranetSwitch.ABUG_ROW3.ABUG_PATH "
call omget using {"dirbackup.tertiarypath"}
"\nExtranetSwitch.ABUG_ROW3.ABUG_INTERVAL "
tint=call omgetnum using {"dirbackup.tertiaryinterval"}
tint=tint/60
tint
"\nExtranetSwitch.ABUG_ROW3.ABUG_USERID "
call omget using {"dirbackup.tertiaryusername"}
"\nExtranetSwitch.ABUG_ROW3.ABUG_PASSWORD "
call omget using {"dirbackup.tertiarypassword"}

// obtain the boot configuration from switch
entry = call omfirst using {"namedconfig"}
cond = (entry != "")

"\nExtranetSwitch.BOOT_SELECT.BOOT_SELECT "
while cond using
{
  call omget using {"namedconfig["entry"].desc"}
  " "
  entry = call omnext using {"namedconfig["entry"]"}
  cond = (entry != "")
}

// obtain performance data
fkey = call omfirst using {"DC.SummaryHistory"}

cond = (fkey != "")
while cond using
{
      svDateString=call omget using {"DC.SummaryHistory["fkey"].Timestamp"}
      svTotalKey=svDateString+"::TOTAL"
      "\nExtranetPerformance.TRENDING "
      svDateString
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].TotalSessions"}
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].AdminSessions"}
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].PPTPSessions"}
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].IPSecSessions"}
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].L2FSessions"}
      " "
      call omget using {"DC.SummaryHistory["svTotalKey"].L2TPSessions"}
      "\nback"

       fkey = call omnext using{"DC.SummaryHistory["fkey"]"}
       cond = (fkey != "")
}
```

```
// obtain the SNMP trap receivers
Error = ""
entry = ""

entry = call omfirst using {"traphost"}
if ( ( entry == "")&& ( Error == "Failure") ) then using
{
  Error = ""
}

if ( entry != "" )then using
{
    cond = (entry != "")
    while cond using
    {
      "\nBayP_ExtranetSNMP.TRAP_TABLE "
      ""entry" "
      " "
      call omget using {"traphost["entry"].enabled"}
      " "
      call omget using {"traphost["entry"].community"}
      "\nback"
      entry = call omnext using {"traphost["entry"]"}
      cond = (entry != "")
    }
}

// obtain the various snmp scripts
Error = ""
filename=call omfirst using {"script"}
if (( filename == "" ) && (Error == "Failure" ) ) then using
{
   Error = ""
}

cond = (filename != "")
while cond using
{
  "\nBayP_ExtranetSNMP.SCRIPT_TABLE "
  //call omget using {"script["filename"].description"}
  file = call omget using {"script["filename"].description"}
  "'"file"'"
  call omget using {"script["filename"].interval"}
  " "
  call omget using {"script["filename"].repeatcount"}
  " "filename""
  "\nback"
  filename = call omnext using {"script["filename"]"}
  cond = ( filename != "" )
}
"\nback"

// obtain the IPX parameters
"\nExtranetIPX.PUB_NET_ADDR.PUB_NET_ADDR "
call omget using {"ipxintfomcls.ipxpublicaddress"}
"\nExtranetIPX.NEAR_SERVER.NEAR_SERVER "
```

```
nearserv = call omget using {"ipxintfomcls.defaultnearestserver"}
" '"nearserv"' "
"\nExtranetIPX.MAX_SAP.MAX_SAP "
call omget using {"ipxintfomcls.sapentries"}
// obtain the IPX interfaces
ifacekey = call omfirst using {"IpxIntfOmCls.IpxPrivateLANS"}
cond = (ifacekey != "")
while cond using
{
    //get values for row
    "\nExtranetIPX.Interface_Table "
    ifacekey
    " "
    call omget using {"IpxIntfOmCls.IpxPrivateLANS["ifacekey"].Slot"}
    " "
    call omget using {"IpxIntfOmCls.IpxPrivateLANS["ifacekey"].Port"}
    " "
    call omget using {"IpxIntfOmCls.IpxPrivateLANS["ifacekey"].IpxAddress"} " "
    call omget using {"IpxIntfOmCls.IpxPrivateLANS["ifacekey"].Encap"}
    " "
    call omget using {"IpxIntfOmCls.IpxPrivateLANS["ifacekey"].Enable"}
    "\nback"

    ifacekey = call omnext using
              {"ipxintfomcls.ipxprivatelans["ifacekey"]"}
    cond = (ifacekey != "")
}
    "\nback"


// obtain RADIUS authentication
Error=""
"\nBayP_RadAuth_Server.ENABLE_RADIUS.ENABLE_RADIUS "
call omget using {"DbRadiusAuthServers.Enabled"}

svAuthKey = call omfirst using {"DbRadiusAuthServers.RadiusAuthServer"}

//if no key, then need to create a server entry in the database
if (svAuthKey == "") then using
{
        Error = ""
      BaseDn = call omget using {"LdapConfig.BaseName"}
      svAuthKey="cn=radius1, ou=Radius, ou=AuthenticationServers,"+BaseDn
      call omcreate using {"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"]"}
}

if(svAuthKey != "") then using
{
 //do gets from database
"\nBayP_RadAuth_Server.DELIMITER.REMOVE_SUFFIX "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].StripUidSuffix"}
"\nBayP_RadAuth_Server.DELIMITER.DELIMITER "
raddel=call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].uidSuffixDelimeter"}
" '"raddel"' "
"\nBayP_RadAuth_Server.ENABLE_AXENT.ENABLE_AXENT "
```

```
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].AuthMethodAXENT"}
"\nBayP_RadAuth_Server.ENABLE_SECURID.ENABLE_SECURID "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].AuthMethodSECURID"}
"\nBayP_RadAuth_Server.ENABLE_CHAP.ENABLE_CHAP "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].AuthMethodCHAP"}
"\nBayP_RadAuth_Server.ENABLE_MSCHAP.ENABLE_MSCHAP "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].AuthMethodMSCHAP"}
"\nBayP_RadAuth_Server.ENABLE_PAP.ENABLE_PAP "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].AuthMethodPAP"}

 //host enable
"\nBayP_RadAuth_Server.ENABLE_PRIMARY.ENABLE_PRIMARY "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].PrimaryHostEnabled"}
"\nBayP_RadAuth_Server.ENABLE_ALT1.ENABLE_ALT1 "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate1HostEnabled "}
"\nBayP_RadAuth_Server.ENABLE_ALT2.ENABLE_ALT2 "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate2HostEnabled"}

 //host names
"\nBayP_RadAuth_Server.PRIM_HOSTNAME.PRIM_HOSTNAME "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].PrimaryHost"}
"\nBayP_RadAuth_Server.ALT1_HOSTNAME.ALT1_HOSTNAME "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate1Host"}
"\nBayP_RadAuth_Server.ALT2_HOSTNAME.ALT2_HOSTNAME "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate2Host"}

 //ports
"\nBayP_RadAuth_Server.PRIM_PORT.PRIM_PORT "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].PrimaryHostPort"}
"\nBayP_RadAuth_Server.ALT1_PORT.ALT1_PORT "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate1HostPort"}
"\nBayP_RadAuth_Server.ALT2_PORT.ALT2_PORT "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate2HostPort"}

 //passwords
"\nBayP_RadAuth_Server.PRIM_SECRET.PRIM_SECRET "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].PrimaryHostPassword"}
"\nBayP_RadAuth_Server.ALT1_SECRET.ALT1_SECRET "
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate1HostPassword"}
"\nBayP_RadAuth_Server.ALT2_SECRET.ALT2_SECRET "
```

```
 call omget using
{"DbRadiusAuthServers.RadiusAuthServer["svAuthKey"].Alternate2HostPassword"}
}


// obtain RADIUS accounting
"\nBayP_RadAcct_Server.ENABLE_INT_RADIUS.ENABLE_INT_RADIUS "
call omget using {"DbRadiusAcctServers.Enabled"}

accsrvkey = call omfirst using {"DbRadiusAcctServers.RadiusAcctServer"}

if ( ( accsrvkey == "" ) && ( Error == "Failure" ) ) then using
{
      Error = ""
}

//if no key, then need to create entry
if (accsrvkey == "") then using
{
      BaseDn = call omget using {"LdapConfig.BaseName"}
      if ( Error == "" ) then using
      {
            accsrvkey="cn=acc1, ou=radius, ou=accounting servers, "+BaseDn
            call omcreate using
{"DbRadiusAcctServers.RadiusAcctServer["accsrvkey"]"}

            if ( Error == "" ) then using
            {
                  accsrvkey=call omfirst using
{"DbRadiusAcctServers.RadiusAcctServer"}
            }
      }
}

//host enable
if ( accsrvkey != "" ) then using
{
    "\nBayP_RadAcct_Server.ENABLE_EXT_RADIUS.ENABLE_EXT_RADIUS "
     call omget using
{"DbRadiusAcctServers.RadiusAcctServer["accsrvkey"].HostEnabled"}

     //host names
    "\nBayP_RadAcct_Server.HOSTNAME.HOSTNAME "
    call omget using {"DbRadiusAcctServers.RadiusAcctServer["accsrvkey"].Host"}

    //ports
    "\nBayP_RadAcct_Server.PORT.PORT "
    call omget using
{"DbRadiusAcctServers.RadiusAcctServer["accsrvkey"].HostPort"}
    "\nBayP_RadAcct_Server.SECRET.SECRET "
    call omGet using
{"DbRadiusAcctServers.RadiusAcctServer["accsrvkey"].HostPassword"}
}

"\nBayP_RadAcct_Server.UPDATE_INTRVL.UPDATE_INTRVL "
call omget using  {"DbGroups.Group[ROOT].Accounts.Account[GENERAL,-
].AcctUpdateFreq"}
```

```
// obtain info on which LDAP we are using

"\nBayP_LDAP.INTERNAL_EXTERNAL "
call omget using  {"LdapConfig.useremote"}

// get info for internal LDAP
"\nBayP_IntLDAP_Server.IS_RUNNING "
call omget using {"Slapd.IsRunning"}
keyLocalAuthServer = call omfirst using {"DbLocalAuthServers.LocalAuthServer"}
"\nBayP_IntLDAP_Server.SUFFIX_ROW.REMOVE_FROM_UID "
ildapui=call omget using
{"DbLocalAuthServers.LocalAuthServer["keyLocalAuthServer"].stripUidSuffix"}
if (ildapui == "") then using
{
   "FALSE"
}
else using
{
  "'"ildapui"'"
}
"\nBayP_IntLDAP_Server.SUFFIX_ROW.DELIMITER "
ildapdel=call omget using
{"DbLocalAuthServers.LocalAuthServer["keyLocalAuthServer"].UidSuffixDelimeter"}
" '"ildapdel"' "

// obtain info for external LDAP
"\nBayP_ExtLDAP_Server.SUFFIX_ROW.REMOVE_FROM_UID "
eldapui=call omget using
{"DbLocalAuthServers.LocalAuthServer["keyLocalAuthServer"].stripUidSuffix"}
if (eldapui == "") then using
{
   "FALSE"
}
else using
{
  "'"eldapui"'"
}
"\nBayP_ExtLDAP_Server.SUFFIX_ROW.DELIMITER "
ldapdel=call omget using
{"DbLocalAuthServers.LocalAuthServer["keyLocalAuthServer"].UidSuffixDelimeter"}
" '"ldapdel"' "
"\nBayP_ExtLDAP_Server.BASE_DN.BASE_DN "
basedn = call omget using {"ldapconfig.remotebasename"}
"'"basedn"'"
"\nBayP_ExtLDAP_Server.MASTER.ELDAP_HOSTNAME "
call omget using {"ldapprofileserver[0].host"}
"\nBayP_ExtLDAP_Server.MASTER.PORT "
call omget using {"ldapprofileserver[0].usessl"}
" "
call omget using {"ldapprofileserver[0].port"}
" "
call omget using {"ldapprofileserver[0].sslport"}
"\nBayP_ExtLDAP_Server.MASTER.BIND_DN "
masterdn = call omget using {"ldapprofileserver[0].bindname"}
"'"masterdn"'"
"\nBayP_ExtLDAP_Server.MASTER.BIND_PASSWORD "
```

```
call omget using {"ldapprofileserver[0].bindpassword"}
"\nBayP_ExtLDAP_Server.SLAVE1.ELDAP_HOSTNAME "
call omget using {"ldapprofileserver[1].host"}
"\nBayP_ExtLDAP_Server.SLAVE1.PORT "
call omget using {"ldapprofileserver[1].usessl"}
" "
call omget using {"ldapprofileserver[1].port"}
" "
call omget using {"ldapprofileserver[1].sslport"}
"\nBayP_ExtLDAP_Server.SLAVE1.BIND_DN "
slave1dn = call omget using {"ldapprofileserver[1].bindname"}
"'"slave1dn"'"
"\nBayP_ExtLDAP_Server.SLAVE1.BIND_PASSWORD "
call omget using {"ldapprofileserver[1].bindpassword"}
"\nBayP_ExtLDAP_Server.SLAVE2.ELDAP_HOSTNAME "
call omget using {"ldapprofileserver[2].host"}
"\nBayP_ExtLDAP_Server.SLAVE2.PORT "
call omget using {"ldapprofileserver[2].usessl"}
" "
call omget using {"ldapprofileserver[2].port"}
" "
call omget using {"ldapprofileserver[2].sslport"}
"\nBayP_ExtLDAP_Server.SLAVE2.BIND_DN "
slave2dn = call omget using {"ldapprofileserver[2].bindname"}
"'"slave2dn"'"
"\nBayP_ExtLDAP_Server.SLAVE2.BIND_PASSWORD "
call omget using {"ldapprofileserver[2].bindpassword"}

cipherkey=call omfirst using {"SslConfig.CipherSpec"}
ccond = (cipherkey != NULL)
while ccond using
{
          "\nBayP_ExtLDAP_Server.Encryption_Table "
       //get values for row
          call omget using {"SslConfig.CipherSpec["cipherkey"].Enabled"}
          " "
          name = call omget using {"SslConfig.CipherSpec["cipherkey"].Name"}
          "'"name"'"
          "\nback"
       cipherkey = call omnext using{"SslConfig.CipherSpec["cipherkey"]"}
          ccond = (cipherkey != "")
}

"\nback"

// Obtain User IP address pool information

"\nBayP_UserIP_Server.ADDR_ACQUIS.ADDR_ACQUIS "
call omget using {"AddressAcquisition.AcquisitionType"}
"\nBayP_UserIP_Server.CACHE.SIZE "
call omget using {"AddressAcquisition.DHCPCacheSize"}
"\nBayP_UserIP_Server.RELEASE.IMMEDIATE "
call omget using {"AddressAcquisition.DHCPReleaseImmediately"}
"\nBayP_UserIP_Server.DHCP.TYPE "
call omget using {"AddressAcquisition.DHCPType"}
"\nBayP_UserIP_Server.PRIMARY_SERVER "
call omget using{"DhcpServer[0].ServerAddress"}
```

```
"\nBayP_UserIP_Server.SECONDARY_SERVER "
call omget using{"DhcpServer[1].ServerAddress"}
"\nBayP_UserIP_Server.TERTIRARY_SERVER "
call omget using{"DhcpServer[2].ServerAddress"}
AddrKey = call omfirst using {"IpAddrPool"}
ccond = (AddrKey != "")
while ccond using
{
  "\nBayP_UserIP_Server.ADDR_TABLE "
  call omget using {"IpAddrPool["AddrKey"].startaddr"}
  " "
  call omget using {"IpAddrPool["AddrKey"].endaddr"}
  " "
  call omget using {"IpAddrPool["AddrKey"].numberofaddrs"}
  " "
  AddrKey
  "\nback"
  AddrKey = call omnext using {"IpAddrPool["AddrKey"]"}
  ccond = (AddrKey != "")
}
  "\nback"

// obtain ethernet interface information


Entry = call omfirst using {"Interface"}
if ( ( Entry == "" ) && ( Error == "Failure" ) ) then using
{
Error = ""
}
CondEntry = (Entry != "")
while CondEntry using
{
    lookType = call omget using {"Interface["Entry"].Type"}

    // only do for lan interfaces
    if (lookType == 2) then using
    {
    "\nExtranetInterface.LAN_Interface "
    call omget using {"Interface["Entry"].slot"}
    " "
    call omget using {"Interface["Entry"].Interface"}
    " "
    " "
    call omget using {"Interface["Entry"].DefaultGateway"}
    " "
    lookLoc = call omget using {"Interface["Entry"].DevLoc"}
    " "lookLoc" "
    call omget using {"Interface["Entry"].Public"}
    " "
    call omget using {"Interface["Entry"].Enabled"}
    " "
    desc = call omget using {"Interface["Entry"].Desc"}
    if ( desc != "" ) then using
    {
      "'"desc"'"
    }
```

```
        else using
        {
          " ' ' "
        }

        ipIntf = call omfirst using {"IpIntf"}
        if ( ( ipIntf == "" ) && ( Error == "Failure" ) ) then using
        {
           Error = ""
        }

        CondipIntf = (ipIntf != "")
        while CondipIntf using
        {
          ipDevLoc  = call omget using {"ipIntf["ipIntf"].DevLoc"}

          if ( ipDevLoc == lookLoc ) then using
          {
              isSystem = call omget using {"ipIntf["ipIntf"].IsSystemIpAddr"}

              // only do for the non-system interfaces
              if ( isSystem == "NO" ) then using
              {
                  //  Keep count of the number of addresses for this card
              " "
              call omget using {"ipIntf["ipIntf"].IpAddr"}
              " "
              call omget using {"IpIntf["ipIntf"].Subnet"}
              " "

              }
           }

          ipIntf = call omnext using {"IpIntf["ipIntf"]"}
          CondipIntf = (ipIntf != "")
        }
        }
       Entry = call omnext using {"Interface["Entry"]"}
       CondEntry = (Entry != "" )
   }

   // obtain WAN information

   Entry = call omfirst using {"Interface"}
   if ( ( Entry == "" ) && ( Error == "Failure" ) ) then using
   {
   Error = ""
   }
   CondEntry = (Entry != "")
   while CondEntry using
   {
       lookType = call omget using {"Interface["Entry"].Type"}
       lookDESC = call omget using {"Interface["Entry"].Hardware"}

       // only do for wan interfaces
       if (lookType == 1) then using
       {
```

```
if (lookDESC != "empty") then using
{
"\nExtranetInterface.WAN_Interface "
call omget using {"Interface["Entry"].slot"}
" "
call omget using {"Interface["Entry"].Interface"}
" "
lookLoc = call omget using {"Interface["Entry"].DevLoc"}
" "lookLoc" "
call omget using {"Interface["Entry"].Enabled"}
" "

desc = call omget using {"PppIntf["lookLoc"].description"}
if ( desc != "" ) then using
{
  "'"desc"'"
}
else using
{
  "' '"
}
" "
locip=call omget using {"PppIntf["lookLoc"].localipaddress"}
" '"locip"' "
ipcp=call omget using {"PppIntf["lookLoc"].ipcpacceptremote"}
if (ipcp == "") then using
{
    "FALSE "
}
else using
{
    "'"ipcp"' "
}
peerip=call omget using {"PppIntf["lookLoc"].peeripaddress"}
" '"peerip"' "
nopap=call omget using {"PppIntf["lookLoc"].NoPapNeg"}
if (nopap == "") then using
{
   "FALSE "
}
else using
{
   "'"nopap"' "
}
nochap=call omget using {"PppIntf["lookLoc"].NoChapNeg"}
if (nochap == "") then using
{
   "FALSE "
}
else using
{
   "'"nochap"' "
}
name = call omget using {"PppIntf["lookLoc"].LocalPapName"}
"'"name"'"
passwd = call omget using {"PppIntf["lookLoc"].LocalPapPasswd"}
" '"passwd"' "
```

```
        noacc=call omget using {"PppIntf["lookLoc"].NoAccNeg"}
        if (noacc == "") then using
        {
           "FALSE "
        }
        else using
        {
           "'"noacc"' "
        }
        nopc=call omget using {"PppIntf["lookLoc"].NoPCNeg"}
        if (nopc == "") then using
        {
           "FALSE "
        }
        else using
        {
           "'"nopc"' "
        }
        lcpfail=call omget using {"PppIntf["lookLoc"].LCPEchoFailure"}
        " '"lcpfail"' "
        lcpint=call omget using {"PppIntf["lookLoc"].LCPEchoInterval"}
        " '"lcpint"' "
        novj=call omget using {"PppIntf["lookLoc"].NoVJNeg"}
        if (novj == "") then using
        {
           "FALSE "
        }
        else using
        {
           "'"novj"' "
        }
        novjc=call omget using {"PppIntf["lookLoc"].NoVJCCompNeg"}
        if (novjc == "") then using
        {
           "FALSE "
        }
        else using
        {
           "'"novjc"' "
        }
        vjslots=call omget using {"PppIntf["lookLoc"].VJMaxSlots"}
        " '"vjslots"' "
        }
        }
        Entry = call omnext using {"Interface["Entry"]"}
        CondEntry = (Entry != "" )
    }
    // obtain PPTP information
    "\nBayP_Tunnel.PPTP "
    call omget using {"DbGroups.Group[ROOT].Accounts.Account[PPTP,-
    ].AuthServerRef2Type"}
    " "
    call omget using {"DbGroups.Group[ROOT].Accounts.Account[PPTP,-
    ].AuthServerRef3Type"}
    "\nBayP_Tunnel.PPTP_MSCHAP_ROW.NOT_ENCRYPT "
    call omget using {"Dbgroups.Group[root].Accounts.Account[PPTP,-
    ].DefaultEncryptionNone"}
```

```
"\nBayP_Tunnel.PPTP_MSCHAP_ROW.MSCHAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[PPTP,-
].DefaultEncryptionRC4_128"}
"\nBayP_Tunnel.PPTP_MSCHAP_ROW.RC440 "
call omget using {"Dbgroups.Group[root].Accounts.Account[PPTP,-
].DefaultEncryptionRC4_40"}
"\nBayP_Tunnel.PPTP_CHAP.PPTP_CHAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[PPTP,-
].DefaultAuthMethodChap"}
"\nBayP_Tunnel.PPTP_PAP.PPTP_PAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[PPTP,-
].DefaultAuthMethodPAP"}

// obtain L2TP information
//"\nBayP_Tunnel.L2TP "
//call omget using {"DbGroups.Group[ROOT].Accounts.Account[L2TP,-
].AuthServerRef2Type"}
//" "
//call omget using {"DbGroups.Group[ROOT].Accounts.Account[L2TP,-
].AuthServerRef3Type"}
"\nBayP_Tunnel.L2TP_MSCHAP_ROW.NOT_ENCRYPT "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2TP,-
].DefaultEncryptionNone"}
"\nBayP_Tunnel.L2TP_MSCHAP_ROW.MSCHAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2TP,-
].DefaultEncryptionRC4_128"}
"\nBayP_Tunnel.L2TP_MSCHAP_ROW.RC440 "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2TP,-
].DefaultEncryptionRC4_40"}
"\nBayP_Tunnel.L2TP_CHAP.L2TP_CHAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2TP,-
].DefaultAuthMethodChap"}
"\nBayP_Tunnel.L2TP_PAP.L2TP_PAP "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2TP,-
].DefaultAuthMethodPAP"}
//"\nback"

// obtain L2F

"\nBayP_Tunnel.L2F "
call omget using {"DbGroups.Group[ROOT].Accounts.Account[L2F,-
].AuthServerRef2Type"} " "
call omget using {"DbGroups.Group[ROOT].Accounts.Account[L2F,-
].AuthServerRef3Type"}

"\nBayP_Tunnel.CHAP_ENABLED.CHAP_ENABLED "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2F,-
].DefaultAuthMethodChap"}
"\nBayP_Tunnel.PAP_ENABLED.PAP_ENABLED "
call omget using {"Dbgroups.Group[root].Accounts.Account[L2F,-
].DefaultAuthMethodPap"}
"\nback"

// obtain IPSec information
  realIPsecKey = ""
  groupKey = call omfirst using {"DbGroups.Group"}
  if ( ( groupKey == "" ) && ( Error == "Failure" ) ) then using
```

```
    {
    // can safely ignore this error
    Error = ""
    }

  if ( groupKey != "" ) then using
    {
    accountKey = call omfirst using
{"DbGroups.Group["groupKey"].Accounts.Account"}
    if ( ( accountKey == "" ) && ( Error == "Failure" ) ) then using
      {
      // can safely ignore this error
      Error = ""
      }

    accountCondition = (accountKey != "") && ( realIPsecKey == "" )
    while accountCondition using
      {
      accountType = call omget using
{"DbGroups.Group["groupKey"].Accounts.Account["accountKey"].TunnelType"}
      if ( accountType == "IPsec" ) then using
        {
        realIPsecKey = accountKey
        }
      accountKey = call omnext using
{"DbGroups.Group["groupKey"].Accounts.Account["accountKey"]"}
      accountCondition = ((accountKey != "") && ( realIPsecKey == "" ))
      } //  while there is a subaccount
    }    // End - groupkey != NULL

"\nBayP_Tunnels.IPSEC "
ref1=call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].AuthServerRef2Type" }
"'"ref1"' "
"\nBayP_TunnelsIP.AUTH_USER.AUTH_USER "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultAuthMethodSHARED_
SECRET" }
"\nBayP_TunnelsIP.AUTH_RSA.AUTH_RSA "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultAuthMethodCERTIFI
CATE_RSA" }
"\nBayP_TunnelsIP.RADAUTH_AXENT.RADAUTH_AXENT "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultAuthMethodAXENT"
}
"\nBayP_TunnelsIP.RADAUTH_SECURITY.RADAUTH_SECURITY "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultAuthMethodSECURID
" }
"\nBayP_TunnelsIP.RADAUTH_GROUP.RADAUTH_GROUP "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultAuthMethodPAP" }
"\nBayP_TunnelsIP.ENCRYP_TRIPLE.ENCRYP_TRIPLE "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultEncryption3DES_MD
5" }
```

```
"\nBayP_TunnelsIP.ENCRYP_ESP56.ENCRYP_ESP56 "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultEncryptionDES_MD5
" }
"\nBayP_TunnelsIP.ENCRYP_ESP40.ENCRYP_ESP40 "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultEncryptionDES_40"
}
"\nBayP_TunnelsIP.ENCRYP_AHSHA.ENCRYP_AHSHA "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultEncryptionHMAC_SH
A" }
"\nBayP_TunnelsIP.ENCRYP_AHMD5.ENCRYP_AHMD5 "
call omget using
{"DbGroups.Group[ROOT].Accounts.Account["realIPsecKey"].DefaultEncryptionHMAC_MD
5" }
"\nBayP_Tunnels.LB_ENABLED.LB_ENABLED "
call omget using {"Loadbalance.Node1Enabled"}
"\nBayP_Tunnels.LB_HOST.LB_HOST "
call omget using {"Loadbalance.Node1"}
"\nBayP_Tunnels.FAILOVER1_ENABLED.FAILOVER1_ENABLED "
call omget using {"Failover.Node1enabled"}
"\nBayP_Tunnels.FAILOVER1_IPADDR.FAILOVER1_IPADDR "
call omget using {"Failover.Node1"}
"\nBayP_Tunnels.FAILOVER2_ENABLED.FAILOVER2_ENABLED "
call omget using {"Failover.Node2enabled"}
"\nBayP_Tunnels.FAILOVER2_IPADDR.FAILOVER2_IPADDR "
call omget using {"Failover.Node2"}
"\nBayP_Tunnels.FAILOVER3_ENABLED.FAILOVER3_ENABLED "
call omget using {"Failover.Node3enabled"}
"\nBayP_Tunnels.FAILOVER3_IPADDR.FAILOVER3_IPADDR "
call omget using {"Failover.Node3"}
"\nback"
```